HAL open science

PACE solver description: DreyFVS

Gabriel Bathie, Gaétan Berthe, Yoann Coudert-Osmont, David Desobry, Amadeus Reinald, Mathis Rocton

To cite this version:

Gabriel Bathie, Gaétan Berthe, Yoann Coudert-Osmont, David Desobry, Amadeus Reinald, et al.. PACE solver description: DreyFVS. Leibniz International Proceedings in Informatics , 2022, 10.4230/LIPIcs.IPEC.2022.31 . hal-03920659

HAL Id: hal-03920659

https://inria.hal.science/hal-03920659

Submitted on 3 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PACE solver description: DreyFVS*

Gabriel Bathie
École Normale Supérieure de Lyon, France

Gaétan Berthe
École Normale Supérieure de Lyon, France

Yoann Coudert–Osmont
Université de Lorraine, CNRS, Inria, LORIA

David Desobry
Université de Lorraine, CNRS, Inria, LORIA

Amadeus Reinald
École Normale Supérieure de Lyon, France

Mathis Rocton
École Normale Supérieure de Lyon, France

Abstract

We describe DreyFVS, a heuristic for Directed Feedback Vertex Set submitted to the 2022 edition of Parameterized Algorithms and Computational Experiments challenge. The Directed Feedback Vertex Set problem asks to remove a minimal number of vertices from a digraph such that the resulting digraph is acyclic. Our algorithm first performs a guess on a reduced instance by leveraging the Sinkhorn-Knopp algorithm, to then improve this solution by pipelining two local search methods.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Directed Feedback Vertex Set, Heuristic, Sinkhorn algorithm, Local search

Supplementary Material *Software (source code)*: `https://doi.org/10.5281/zenodo.6638217`

1 Introduction

In the following, a directed graph, or digraph $G = (V, E)$ is a set of vertices V, and a set of arcs E, consisting of ordered pairs of vertices. For a vertex $v \in V(G)$, we denote its out-neighbourhood by $N^+(v) = \{u \in V(G) : (v, u) \in E\}$, and its in-neighbourhood by $N^-(v) = \{u \in V(G) : (u, v) \in E\}$. A directed cycle is a sequence $(u_1, ..., u_k, u_1)$ such that $(u_i, u_{i+1[k]}) \in E(G)$. A directed acyclic graph, DAG for short, is a digraph containing no cycles. Given X a set of vertices, we let $G[X]$ be the graph induced by X. A strongly connected component of G is a vertex set X such that there is a path between any pair of vertices in $G[X]$. We say that (u, v) is a *digon* if it is a (directed) cycle of length two. We say that a set $C \subseteq V$ is a d-clique whenever $|C| = d$ and each pair $(c, c') \in C$ forms a digon. The Directed Feedback Vertex Set problem takes as input $G = (V, E)$ and asks for a set $X \subseteq V$ such that $G[V \backslash X]$ contains no directed cycle. We consider the minimization version of the problem, asking for X to be of minimal size.

Our algorithm first applies reduction rules to obtain a smaller instance, guesses a first solution using a Sinkhorn-Knopp algorithm, then uses two heuristics to improve the solution in a local manner. A C++ implementation is available on a public git repository [1].

* This is a brief description of one of the highest ranked solvers of PACE Challenge 2022. It has been made public for the benefit of the community and was selected based on the ranking. PACE encourages publication of work building on the ideas presented in this description in peer-reviewed venues.

In Section 2, we present the steps needed to guess our first solution. We first describe our reduction rules 2.1, based on those described by Lin and Jou [4], and provide an extension to their CORE rule. Then, we give an overview of the Sinkhorn-Knopp method in 2.2. In Section 3, we describe our two local search heuristics.

2 Guessing a solution

2.1 Reductions

At any step of the reduction, one can deal with each strongly connected component independently, as solutions to the instance are exactly unions of solutions to each strongly connected component. The main reductions rules applied in our algorithm are those developed in [4], namely LOOP, IN0, IN1, OUT0, OUT1, PIE, CORE and DOME. Given a vertex $v \in V(G)$, a bypass of v consists in the deletion of v, followed by the addition of arcs (u, w) for any $u \in N^-(v), w \in N^+(v)$. The goal of the bypass operation is to reduce instances where v can always be replaced by another vertex in the solution. Towards motivating our extension of the CORE rule, we first describe rule IN1, note that the OUT1 rule is symmetric when v admits a single out-neighbour.

▶ **Reduction Rule 1** (IN1). *If a vertex $v \in V(G)$ admits a single in-neighbour, bypass v.*

Indeed, the in-neighbour of v is part of any cycle containing v. A solution using v in the original graph yields a solution of the same size using its in-neighbour in the reduced graph. Thus, a solution in the reduced graph is also a solution in the original one.

In [4], the CORE rule looks to bypass a vertex v that is a CORE of some d-clique, meaning that it forms a d-clique with all its (in or out) neighbours. We extend this rule by applying it whenever its in-neighbours or its out-neighbours form a d-clique.

▶ **Reduction Rule 2** (CORE'). *If a vertex $v \in V(G)$ is such that $N^-(v)$ or $N^+(v)$ forms a d-clique, bypass v.*

Indeed, considering the case where $N^-(v)$ forms a d-clique, any solution to DFVS must contain at least one vertex per digon in $N^-(v)$, thus at least $d-1$ vertices of $N^-(v)$ in total. Then, by the same arguments as rule IN1, taking the remaining in-neighbour of v in a solution is at least as good as taking v. The case is symmetrical when $N^+(v)$ forms a d-clique, using the OUT1 rule.

2.2 The Sinkhorn–Knopp Algorithm

Given a graph G, we construct our first solution X by the heuristic of Shook and Beichl [5]. We call a set of vertex disjoint cycles a disjoint cycle union, or DCU for short. Given an $n \times n$ matrix A, the permanent of A is defined as $perm(A) = \sum_{\sigma \in \mathfrak{S}_n} \prod_{i=1}^{n} a_{i\sigma(i)}$. The idea of the heuristic is to iteratively add vertices v that are contained in many DCUs to the solution X. While computing all DCUs is not efficient, a first observation is that the permanent of A allows us to count the number of spanning DCUs in G, where a spanning DCU is a DCU covering all vertices of the graph. To lift the spanning constraint and count all DCUs, it is possible to take the permanent of the matrix $M = A + I$. If we denote by M_{ij} the matrix M with row i and column j removed, we define the *m-balance* matrix m-bal(M) as in [2]:

$$\text{m-bal}(M)_{ij} = \frac{m_{ij} \cdot perm(M_{ij})}{perm(M)},$$

Then, m-bal$(M)_{vv}$ is the fraction of DCUs that do not contain v. Thus, small values lead us to consider the addition of v to the solution. Computing m-bal(M) is still hard, nevertheless [2] provides a way to approximate m-bal(M). This is done by computing the Sinkhorn balance matrix s-bal(M) through the Sinkhorn-Knopp algorithm [6] applied on M. Sinkhorn-Knopp is an iterative algorithm where each iteration has a $\mathcal{O}(|E|)$ complexity. We decided to apply $\log|V|$ iterations leading to a $\mathcal{O}(|E|\log|V|)$ complexity for the computation of s-bal(M). We can now describe our iterative construction of solution X. As long as graph G is not empty, we compute s-bal(M) and add the vertex v minimising s-bal$(M)_{vv}$ to X, then, we apply our reduction rules to the remaining graph $G \setminus \{v\}$ and re-iterate.

3 Local search

The solution obtained in the previous section is further improved using two greedy local search heuristics sequentially. In the following, G refers to a strongly connected component of the reduced instance, and X to our current solution for DFVS.

3.1 Vertex Swapping

The first local operation consists of a simple vertex swap. Namely, we remove a vertex from the current solution X, and if a cycle is created we attempt to replace it with another vertex of the cycle at random. We iterate through every vertex v of X. If $G \setminus X \cup \{v\}$ is acyclic, we remove v from X such that the solution size decreases by 1. Otherwise, we consider a cycle C in $G \setminus X \cup \{v\}$, and choose some other vertex $w \in C \setminus \{v\}$, replacing v with w in X if $G \setminus X \cup \{v\} \setminus \{w\}$ is acyclic.

3.2 Vertex Ordering Perturbation

Recall that a digraph is acyclic if and only if its set of vertices admits a topological ordering, that is, an order π over V such that for every arc $(u,v), \pi(u) < \pi(v)$. Our idea, similar to that of [3], is to define a unique DFVS X_π from any ordering π of V, and then swap the positions of random pairs of vertices in π, aiming to reduce the size of the solution X_π.

We first show how to construct X_π from π. Start with $X_\pi^0 = \emptyset$, and then define X_π^n inductively as follows:

$$X_\pi^{n+1} = \begin{cases} X_\pi^n & \text{if } N^+(v_\pi^{n+1}) \cap V_\pi^n \subseteq X_\pi^n \\ X_\pi^n \cup \{v_\pi^{n+1}\} & \text{otherwise} \end{cases}$$

Where $V_\pi^n = \{v \in V \mid \pi(v) \leqslant n\}$ and v_π^n is the vertex such that $\pi(v_\pi^n) = n$. We then set $X_\pi = X_\pi^{|V|}$. Notice that X_π can be computed in linear time using a traversal of G in the order given by π.

We can use this structure to derive a simple local search algorithm, which repeatedly chooses a uniformly random pair of vertices (u,v) and swaps their positions to obtain a new ordering π' such that $\pi'(u) = \pi(v)$, $\pi'(v) = \pi(u)$ and $\pi'(w) = \pi(w)$ for $w \notin \{u,v\}$. We then compute $|X_{\pi'}|$, and if $|X_{\pi'}| \leqslant |X_\pi|$, we validate the swap and set π to π', otherwise we cancel the swap and leave π unchanged. In practice, iterating this process quickly reduces the size of the solution X_π on most instances.

Given a solution X obtained after applying the local search of the previous section 3.1, we can compute the initial value of π using a topological sort of the acyclic digraph $G[V \setminus X]$, to which we append a random permutation of X.

References

1 Gabriel Bathie, Gaétan Berthe, Yoann Coudert-Osmont, David Desobry, Amadeus Reinald, and Mathis Rocton. DreyFVS, 6 2022. URL: https://github.com/Nanored4498/DreyFVS, doi:10.5281/zenodo.6638217.

2 Isabel Beichl and Francis Sullivan. Approximating the permanent via importance sampling with application to the dimer covering problem. *Journal of Computational Physics*, 149(1):128–147, 1999. doi:10.1006/jcph.1998.6149.

3 Philippe Galinier, Eunice Lemamou, and Mohamed Wassim Bouzidi. Applying local search to the feedback vertex set problem. *Journal of Heuristics*, 19(5):797–818, oct 2013. doi:10.1007/s10732-013-9224-z.

4 Hen-Ming Lin and Jing-Yang Jou. On computing the minimum feedback vertex set of a directed graph by contraction operations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(3):295–307, 2000. doi:10.1109/43.833199.

5 James Shook and Isabel Beichl. Matrix scaling: A new heuristic for the feedback vertex set problem, June 2014. URL: https://math.nist.gov/mcsd/Seminars/2014/2014-06-10-Shook-presentation.pdf.

6 Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.